

# ANASIEZE IKENNA – CLOUD & AI ENGINEER

## Project: Cloud SQL Data Pipeline & Analytics Project

### Overview

This guide provides a comprehensive walkthrough of a cloud-native data engineering project built on Google Cloud Platform (GCP).

For technical teams: Detailed commands, code snippets, and architectural diagrams.

For non-technical teams: Business context, impact analysis, and visual workflows.

### Business Problem

Airlines and transportation agencies need to analyze flight performance, delays, and operational efficiency across 300,000+ flight records to optimize routes, reduce costs, and improve passenger experience.

### Technical Solution

A scalable ETL pipeline that:

1. Extracts flight data from public sources.
2. Transforms and loads it into Cloud SQL (PostgreSQL).
3. Analyses data using SQL to generate actionable insights.

### Architecture Diagram

[Cloud Storage: Raw CSV Files]

↓

[Cloud SQL: Data warehouse]

↓

[SQL Analytics: Query]

↓

[Dashboards & Reports]

### Step-by-Step Setup

#### ⇒ Prerequisites

- Google Cloud Platform account
- Basic knowledge of SQL and command line
- Access to Cloud Shell

#### ⇒ Environment Setup

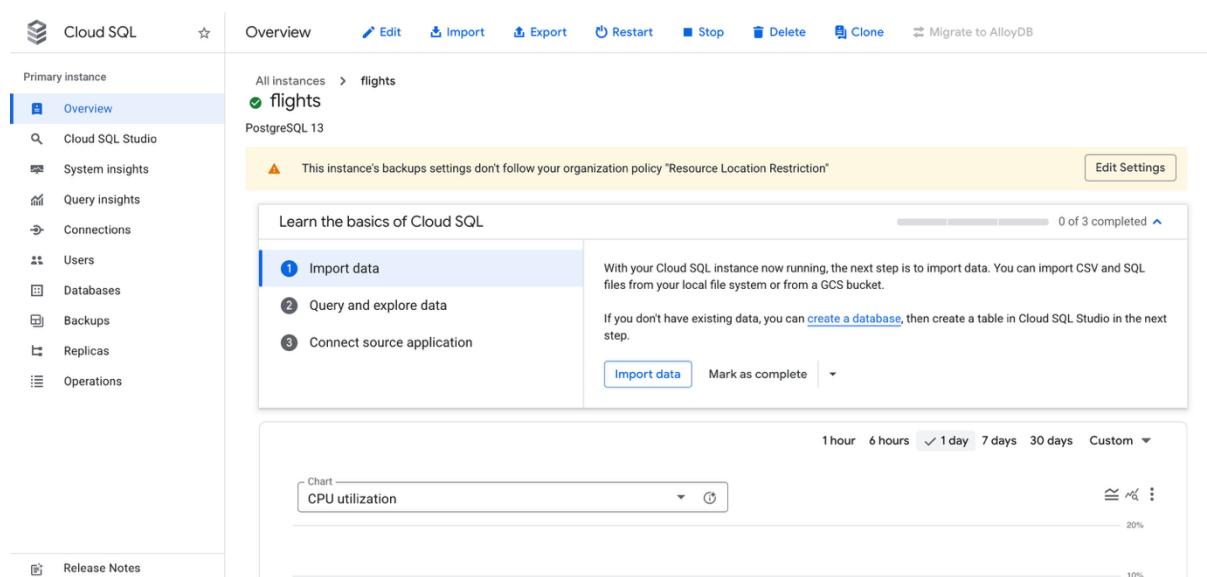
Step 1: Launch Cloud Shell

1. Log into [Google Cloud Console](<https://console.cloud.google.com>).

## 2. Click the Cloud Shell icon.

### Step 2: Create Cloud SQL Instance

```
```bash
gcloud sql instances create flights \
--database-version=POSTGRES_13 \
--cpu=2 \
--memory=8GiB \
--region=us-central1 \
--root-password="Your secure password"
````
```



The screenshot shows the Google Cloud Platform Cloud SQL Overview page for a 'flights' instance. The instance is a PostgreSQL 13 database. A yellow warning box at the top right states: 'This instance's backups settings don't follow your organization policy "Resource Location Restriction"'. Below this, a 'Learn the basics of Cloud SQL' section lists three steps: 'Import data', 'Query and explore data', and 'Connect source application'. The 'Import data' step is highlighted. A 'Chart' section shows 'CPU utilization' over the last 24 hours, with a 20% utilization peak. The left sidebar includes links for Overview, Edit, Import, Export, Restart, Stop, Delete, Clone, and Migrate to AlloyDB.

### Non-Technical Explanation:

This command provisions a managed PostgreSQL database with 2 CPUs and 8GB RAM, located in Iowa (us-central1). It's like renting a secure, scalable database server in the cloud.

### Step 3: Prepare Cloud Storage

```
```bash
Create a unique bucket for data files
export PROJECT_ID=$(gcloud info --format='value(config.project)')
export BUCKET=${PROJECT_ID}-flight-data
````
```

### Non-Technical Explanation:

Creates a cloud storage folder (bucket) to hold raw flight data files.

## ⇒ ETL Pipeline Development

Data Extraction  
For Technical Teams

```
```bash
Download sample flight data
wget https://storage.googleapis.com/cloud-training/OCBL013/nycflights13.csv
Upload to Cloud Storage
gsutil cp nycflights13.csv gs://$BUCKET
```

```

For Non-Technical Teams

- Data is sourced from the US Bureau of Transportation Statistics.
- Files are automatically transferred to secure cloud storage.

⇒ **Data Loading into Database**

Step 1: Create Table Schema

Technical Code:

```
```sql
-- create_table.sql
CREATE TABLE flights (
  "Year" TEXT,
  "Quarter" TEXT,
  "Month" TEXT,
);
```

```

Step 2: Import Data

```
```bash
Import CSV into PostgreSQL
gcloud sql import csv flights \
gs://$BUCKET/nycflights13.csv \
--database=postgres \
--table=flights
```

```

|   |   |                  |
|---|---|------------------|
|  | <a href="#">Importing from <u>201501.csv</u> to <u>flights</u></a>      | 0 min 19 sec     |
|  | <a href="#">Imported from <u>create_table.sql</u> to <u>flights</u></a> | 3:21:47 PM GMT-3 |
|  | <a href="#">Edited <u>flights</u></a>                                   | 3:17:53 PM GMT-3 |

Non-Technical Explanation:

This step loads the flight records into the database table, similar to importing a spreadsheet into a more powerful analysis tool.

⇒ **Database Analytics**

## Key Business Queries

### Query 1: Top 3 Busiest Airports

```
```sql
SELECT "Origin", COUNT() AS num_flights
FROM flights
GROUP BY "Origin"
ORDER BY num_flights DESC
LIMIT 3;
```
```

Output:

| Origin | Num_flights |
|--------|-------------|
| ATL    | 29,512      |
| ORD    | 23,484      |
| DFW    | 23,153      |

```
Connecting to database with SQL user [postgres].Password:
psql (16.11 (Ubuntu 16.11-1.pgdg24.04+1), server 13.23)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=> \c bts;
Password:
psql (16.11 (Ubuntu 16.11-1.pgdg24.04+1), server 13.23)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
You are now connected to database "bts" as user "postgres".
bts=> SELECT "Origin", COUNT(*) AS num_flights
FROM flights GROUP BY "Origin"
ORDER BY num_flights DESC
LIMIT 5;
Origin | num_flights
-----+-----
ATL   |    29512
ORD   |    23484
DFW   |    23153
LAX   |    17340
DEN   |    17090
(5 rows)

bts=> █
```

### Query 2: Average Delay by Airline

```
```sql
SELECT "Reporting_Airline",
       AVG("DepDelayMinutes") AS avg_delay
FROM flights
GROUP BY "Reporting_Airline"
ORDER BY avg_delay DESC;
```
```

Business Insight: Identifies airlines with frequent delays for operational reviews.

- Cloud & AI Engineer: Anasieze Ikenna
- Email: ikenna.anasieze@gmail.com